mfms

Сервис доставки Push-сообщений

Руководство по интеграции библиотеки PushLib 2.0 Lite
в приложение на базе ОС Android

Оглавление

Термины и сокращения

Термин	Полная форма	Описание
appPackage		Уникальный код приложения заказчика в магазинах приложений Google Play и App Store. Используется при регистрации приложения на push-сервере
deviceUid		Уникальный идентификатор мобильного приложения, установленного на конкретном устройстве. Формируется таким образом, что является постоянным при обновлениях и переустановках приложения на конкретном аппарате клиента
deviceAddress		Уникальный адрес мобильного приложения, установленного на конкретном устройстве. Формируется по заданному алгоритму при каждом запуске приложения на стороне Push-сервера. Может меняться при изменении Push-адреса приложения в PNS. Это зависит от настроек, заданных для приложения на Push-сервере.
PNS	Push Notification Services	Провайдеры Push-уведомлений APNS (Apple Push Notification Service) и GCM (Google Cloud Message). Обеспечивают доставку Push-уведомлений в приложение на устройство клиента.
Push-адрес		Уникальный адрес приложения в PNS, может изменять значение по инициативе PNS
Push-сервер		Программно-аппаратный комплекс, который реализует сервис доставки Push-сообщений. В частности, механизм регистрации приложений на Push-сервере, отправку Push-уведомлений и доставку содержимого Push-сообщений в приложение на устройстве, резервирование доставки с помощью sms-сообщений, хранение статусов доставки Push-сообщений
Push-сообщение		Сообщение от заказчика, включающее Push-уведомление и содержимое сообщения в текстовом формате. В качестве содержимого заказчик может передавать: тексты, а также бинарные данные в формате Base64
Push-уведомление		Короткое уведомление, которое Push-сервер отправляет в приложение посредством сервисов PNS. Push-уведомления, доставленное в приложение, инициирует процедуру получения содержимого Push-сообщения с Push-сервера
Библиотека	Push-библиотека	Компонент мобильного приложения, участвует в интеграции с приложением. Обеспечивает регистрацию приложения на Push-сервере, привязку дополнительных идентификаторов, доставку Push-уведомлений и содержимого Push-сообщений в приложение
Заказчик		Издатель мобильного приложения, является инициатором отправки Push-сообщений
Приложение	Мобильное приложение	Программное обеспечение заказчика, установленное на мобильном устройстве клиента, в которое интегрирована Push-библиотека

Введение

Документ представляет собой руководство разработчика, описывающее порядок работы по интеграции Push-библиотеки Lite в приложение заказчика. В документе приводится необходимая информация по интеграции, настройке и администрированию библиотеки.

Назначение и область применения

Назначение

Библиотека предназначена для решения следующих задач в рамках сервиса доставки Push-сообщений:

- регистрация приложения на Push-сервере;
- доставка Push-уведомлений и Push-сообщений в приложение;
- подтверждение доставки уведомлений на Push-сервере;
- отправка файлов и сообщений из приложения;
- отслеживание статусов сообщений.

Область применения

Процедура интеграции библиотеки в приложение заказчика является частью процесса внедрения сервиса доставки Push-сообщений.

Интеграция библиотеки в приложение

Требования к интеграции

Библиотека поддерживает приложения на базе ОС Android версии 4.0 и выше. Библиотека поддерживает работу с PNS-сервисом Firebase Cloud Messaging (FCM).

Порядок интеграции библиотеки

Интеграция библиотеки в приложение производится в следующей последовательности:

1. Обновить файл `AndroidManifest.xml`.
2. Добавить зависимости в файлы `build.gradle`.
3. Реализовать классы-наследники `PushBroadcastReceiver` и `PushServerIntentService`.
4. Инициализировать `Firebase`.
5. Написать код инициализации библиотеки.
6. Настроить сборку приложения в консоли администрирования.

Обновление файла AndroidManifest.xml

Для настройки интеграции библиотеки в приложение необходимо обновить файл `AndroidManifest.xml`, добавив в него приведенный ниже код:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    <permission
        android:name="="[YOUR_APPLICATION_PACKAGE_NAME].permission.pushserver.RECEIVE"
    android:protectionLevel="signature" />
```

```
    <uses-permission
android:name="="[YOUR_APPLICATION_PACKAGE_NAME].permission.pushserver.RECEIVE" />
     <application>

    <!-- Setting up Push Lib  -->
      <meta-data
           android:name="com.pushserver.android.service"
android:value="[YOUR_APPLICATION_PACKAGE_NAME].[IMPLEMENTATION_INTENT_SERVICE]" />
      <service
android:name="[YOUR_APPLICATION_PACKAGE_NAME].[IMPLEMENTATION_INTENT_SERVICE]" />

     <receiver
           android:name="[YOUR_BROADCAST_RECEIVER]"
           android:exported="false"

     android:permission="[YOUR_APPLICATION_PACKAGE_NAME].permission.pushserver.RECEIVE">
           <intent-filter>
                <action android:name="com.pushserver.android.NEW_PUSH_MESSAGE_EVENT" />
                <action android:name="com.pushserver.android.RECEIVER_CHANGED_EVENT" />
                <action
android:name="com.pushserver.android.DEVICE_ADDRESS_CHANGED_EVENT" />
                <action android:name="com.pushserver.android.HAS_SECURED_MESSAGE" />
                <action android:name="com.pushserver.android.ERROR_EVENT" />
           </intent-filter>
     </receiver>
     </application>
</manifest>
```

Используемые параметры:

- [YOUR_BROADCAST_RECEIVER] – путь к классу-наследнику `PushBroadcastReceiver`, который будет обрабатывать события библиотеки;
- [YOUR_APPLICATION_PACKAGE_NAME] – полное наименование пакета приложения;
- [IMPLEMENTATION_INTENT_SERVICE] – имя класса, реализующего `ServerIntentService`.

Добавление зависимостей в Gradle

Необходимо обновить корневой `build.gradle` и файлы `build.gradle` приложения, добавив следующие зависимости:

1. Корневой `build.gradle`:

```
allprojects {
    repositories {
        google()
        jcenter()
        maven {
            url 'https://maven-pub.mfms.ru/repository/maven-public/'
        }
    }
}
```

2. `build.gradle` приложения:

```
dependencies {
   debugImplementation 'com.mfms.android:push-lite-debug:2.6.20'
   releaseImplementation 'com.mfms.android:push-lite-release:2.6.20'
}
```

Важно! Debug и Release версия библиотеки используют разные сервера, поэтому для сборок приложений, которые выкладываются в Google Play Market, необходимо использовать строго push-lite-release.

Классы для реализации

Для интеграции библиотеки в приложение требуется реализовать следующие классы-наследники от:

- `PushBroadcastReceiver` – обработчик событий библиотеки;
- `PushServerIntentService` – обработчик сохранения сообщений.

Обработчик событий библиотеки PushBroadcastReceiver

Для обработки событий библиотеки в наследнике класса `PushBroadcastReceiver` требуется реализовать методы, представленные в Таблице 1.

Таблица 1. Методы класса PushBroadcastReceiver

onNewPushNotification

Метод вызывается при получении нового `PushNotification` от FCM. В качестве параметра передается текст, полученный из `shortMessage` при отправке PNS (также Bundle с содержимым Push-сообщения). Необходимо отобразить локальное уведомление в области уведомлений ОС.

Метод:

```
protected abstract void onNewPushNotification
(
  android.content.Context context
  java.lang.String alert
  android.os.Bundle bundle
)
```

Параметры:

context – контекст

alert – текст PushNotification

bundle – содержимое Push-сообщения

onStatusChanged

Метод вызывается при изменении Push-адреса в FCM.

Метод:

```
protected abstract void onStatusChanged
(
  android.content.Context context,
  java.lang.String registrationId
)
```

Параметры:

context – контекст

registrationId – идентификатор приложения в Push-сети

onDeviceAddressChanged

Метод вызывается после изменения значения deviceAddress мобильного устройства.

Метод:

```
protected abstract void onStatusChanged
(
  android.content.Context context,
  java.lang.String registrationId
)
```

Параметры:

context – контекст

newDeviceAddress – новое значение `deviceAddress`

onDeviceAddressProblems

Метод вызывается при наличии проблем, препятствующих генерации `deviceAddress`, и получает код проблемы:

- NO_GCM_ID – в случае отсутствия `registrationId Google`;
- NO_CONTEXT – неправильная инициализация библиотеки, не передан контекст приложения;
- NO_DEVICE_ID – не установлен идентификатор приложения;
- BAD_ENCODING – отсутствует поддержка кодировки UTF-8 в приложении.

Метод:

```
protected abstract void onDeviceAddressProblems
(
  android.content.Context context,
  java.lang.String errorCode
)
```

Параметры:

context – контекст

errorCode – код ошибки

onError

Метод вызывается при возникновении ошибки взаимодействия с сервером.

Метод:

```
protected abstract void onError
(
  android.content.Context context,
  java.lang.String errorCode)
```

Параметры:

context – контекст

errorCode – код ошибки

Обработчик событий библиотеки PushServerIntentService

Для получения списка сообщений с Push-сервера в наследнике класса `PushServerIntentService` требуется реализовать методы, представленные в Таблице 2.

Таблица 2. Методы класса PushServerIntentService

saveMessages

Метод вызывается при получении списка новых PushMessages с Push-сервера.

Метод:

```
protected abstract boolean saveMessages(final List<PushMessage> pushMessages);
```

Параметры:

pushMessages – список сообщений.

Возвращаемые значения:

true – при успешном сохранении сообщений, иначе *false.*

messagesWereRead

Метод вызывается при получении списка идентификаторов прочитанных Push-сообщений.

Метод:

```
protected abstract void messagesWereRead(final List<String> messageIds);
```

Параметры:

messageIds – список идентификаторов прочитанных Push-сообщений.

Примечание. Метод `saveMessages` вызывается не в main-thread.

Структура сообщения PushMessage

Структура сообщения `PushMessage` (является параметров в методе `saveMessages`) представлена в Таблице 4.

Таблица 3. Структура сообщения PushMesage

Имя	Тип	Значение
messageId	String	Идентификатор сообщения
sentAt	String	Дата и время формирования сообщения в формате DD.MM.YYYY hh:mm:ss Z, где Z – часовой пояс в формате RFC 822
shortMessage	String	Сокращенная форма сообщения. Отправляется через Push-сеть в виде системного Push-уведомления (Alert). Предназначено для отображения в списке сообщений в виде заголовка
fullMessage	String	Полный текст Push-сообщения (base64-encoded HTML)
secured	Boolean	Признак безопасного сообщения
read	Boolean	Статус прочтения сообщения
sessionKey	String	Сессионный ключ. Используется библиотекой для аутентификации устройства на Push-сервер

Пример сообщения:

```
{messageId: "10",
sentAt: "24.10.2012 12:41:57",
```

```
shortMessage: "Уважаемый, ФИО! Изменились …",
fullMessage: "0KPQstCw0LbQsNC10LzRi9C5INC60LvQuNC10L3RgiEg0JjQt",
secured: true
read: true
sessionKey: sessionKey
}
```

Класс PushController для реализации методов библиотеки

Класс `PushController` реализует методы для предоставления доступа к функциональности библиотеки. Описание методов представлено в Таблице 3.

Таблица 4. Методы класса PushController

Метод	Описание
getDeviceUid()	Метод возвращает текущий `deviceUid`
setDeviceUid(java.lang.String deviceUid)	Метод устанавливает внешнее значение для `deviceUid`. Применяется, если по каким-либо причинам встроенный механизм генерации `deviceUid` не подходит заказчику
sendMessage(String context, boolean isSystem)	Синхронная отправка (входящего) сообщения на Push-сервер. isSystem = true, если сообщение системное (например typing)
sendMessageAsync(String context, boolean isSystem)	Асинхронная отправка (входящего) сообщения на Push-сервер. isSystem = true, если сообщение системное (например, typing)
resetCounterSync()	Сбрасывает счётчик непрочитанных сообщений
notifyMessageUpdateNeeded()	Метод для принудительной загрузки сообщений с сервера
notifyMessageRead(final String messageId)	Отмечает сообщение с указанным идентификатором как прочитанное
getDeviceAddress()	Получает текущий `deviceAddress`
getVersion()	Получает текущую версию библиотеки

Инициализация Firebase

Для инициализации Firebase необходимо воспользоваться соответствующим ассистентом. Для этого нужно зайти в пункт меню `Android Studio -> Tools -> Firebase -> Cloud Messaging -> Set up Firebase Cloud Messaging`. В открывшейся панели необходимо нажать кнопку `Add FCM to your app` и принять изменения. Это добавит необходимые зависимости в ваш проект.

Кроме этого необходимо поместить в корневую папку приложения (рядом с `build.gradle` приложения) файл `google-services.json`. Файл можно сформировать самостоятельно в консоли Firebase или запросить у своего консультанта по внедрению.

Код инициализации библиотеки

Для инициализации библиотеки необходимо вызвать метод `PushController.getInstance(this).init()`. Крайне желательно делать это при запуске приложения.

Библиотека выполнит вызов callback-метода `onDeviceAddressChanged`. После этого станет доступна процедура отправки Push-сообщений на мобильное устройство по его `deviceAddress`.

Настройка сборки приложения

Вместе с этим руководством предоставляется доступ в тестовую консоль администрирования Push-сервера, который необходим, чтобы проверить корректность интеграции библиотеки в приложение.

Консоль администрирования позволяет:

- настраивать сборки приложений;
- просматривать список устройств;
- отправлять тестовые Push-сообщения в свое мобильное приложение;
- смотреть их статусы доставки.

Перед запуском приложения необходимо настроить сборку приложения (вы можете попросить сделать это у вашего консультанта по внедрению). Для этого необходимо в консоли администрирования зайти в меню «Приложения» и нажать «+» и заполнить следующие поля (далее приведены значения полей для тестовых целей):

- Платформа Push Notification Service – GCM;
- Google SenderId – идентификатор отправителя, полученный из консоли Firebase;
- Google API Key – ключ доступа, используемый для отправки Push, полученный из консоли Firebase;
- Использовать API FCM при отправке – Включено;
- Отмена резервной SMS (Upstream Message) – Отключено;
- Отмена резервной SMS (Delivery Reciepts) – Отключено;
- Количество попыток переотправки при ошибке – 2;
- Значение badge – Количество непрочитанных;
- Блокируемые версии ОС (regexp) – Оставить пустым;
- Идентификатор Push-сервера ID – Оставить пустым;
- Автоматическое подтверждение доставки при скачивании контента – Отключено;
- Отправлять Push даже если истек StopTime – не отправлять;
- Предельный TTL для быстрой отправки – 30;
- Динамические таймауты отправки резервных SMS – оставить пустым;
- Ключи доступа – необходимо добавить ключ с UID = ProviderUID из библиотеки (фигурирует в logcat при старте приложения).

Коды ошибок

В метод, обрабатывающий ошибки взаимодействия с Push-сервером, передаются коды ошибок. Описание кодов ошибок представлено в Таблице 5.

Таблица 5. Коды ошибок

Код ошибки	Описание
DEVICE_ADDRESS_INVALID	Указано некорректное значение идентификатора `deviceAddress`
ACCESS_DENIED	Недостаточно прав для выполнения операции
INTERNAL_SERVER_ERROR	Внутренняя ошибка сервера: требуется повторить запрос позже
BAD_PARAMETERS	Указаны неверные параметры запроса. Данная ошибка возникает в случае неверного значения `deviceAddress` или его отсутствия на Push-сервере
IO_ERROR	Сетевая ошибка
JSON_FORMING_ERROR	Неверно сформировано JSON-сообщение

Примечание! При возникновение ошибок сетевого взаимодействия или некорректной передаче параметров методам значение `errorCode` > 0. Если ошибок нет, тогда значение `errorCode` = nil, либо `errorCode` = 0; `errorCode:Int32` = 0. Описание ошибки: `errorDescription`.

Дополнительные настройки

В `Android.manifest` можно также задать некоторые дополнительные настройки (значения которых установлены по умолчанию). Рекомендуется делать это только по согласованию с вашим консультантом по внедрению. Список таких настроек приведен в Таблице 6.

Таблица 6. Дополнительные настройки

Код настройки	Описание
com.pushserver.android.providerUid	Ключ доступа. По умолчанию является функцией `App_package`. Аналогичное значение нужно задать на сервере
com.pushserver.android.logs	Признак необходимости логирования. По умолчанию true – для debug версии и false – для release версии
com.pushserver.android.heartbeatInterval	Интервал опроса FCM (в миллисекундах). По умолчанию 300 000 (5 минут)
com.pushserver.android.serverUrlList	Список адресов Push-серверов. По умолчанию для debug версии: { http://pushservertest.mfms.ru/push-test, https://pushserver.mfms.ru/banks } и для release версии: { https://pushserver.mfms.ru/banks }
com.pushserver.android.serverIdList	Список идентификаторов Push-серверов. По умолчанию {1, 3} – для debug версии и { 3 } – для release версии
com.pushserver.android.serverPrimaryId	Идентификатор основного сервера. По умолчанию – 1 для debug версии и 3 – для release версии
com.pushserver.android.useDefaultDeviceUID	Необходимость автоматической генерации `deviceUid`. По умолчанию true
com.pushserver.android.appPackagePostfix	Строковая настройка которая используется для изменения поведения BackEnd в новой версии (например для смены senderId в Firebase)